VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Záznamník sériového přenosu na RS232

Serial Comunication Logger

2009 Marek Pszczolka

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 21. srpna 2009 .....

Rád bych na tomto místě poděkoval panu Ing. Michalu Krumniklovi, protože bez jeho pomoci by tato práce nevznikla.

Abstrakt

Tato bakalářská práce je zaměřena na záznam dat příchozích ze sériové linky RS232. Tyto data jsou poté ukládána na libovolné paměťové médium (v tomto případě SD karta). Cílem je navrhnout hardwarovou jednotku a obslužný software, které umožní záznam dat na již zmíněné paměťové médium. Tyto data je pak možno přenést a zobrazit na počítači. Teoretická část je věnována použitým hardwarovým částem a praktická část popisuje konkrétní realizaci na mikropočítači Atmel ATmega32. Obslužný software je ve formě knihoven v programovacím jazyce C.

Klíčová slova: RS232, SD karta, Atmel ATmega32, programovací jazyk C

Abstract

This bachelor thesis is oriented on data logging that arrivals from serial interface RS232. These data are saved to any memory medium (in these case it is an SD card). The main goal of this project is to develop a hardware unit and utility software, which will make possible data logging to already mentioned memory medium. It is possible to transfer these data and display on regular computer. Theroretical part is devoted to hardware components used in this projekt and the practical part describes concrete implementation at microcontroler Atmel ATmega32. The utility software is in libraries, which have been written in programming language C.

Keywords: RS232, SD card, Atmel ATmega32, programming language C

Seznam použitých zkratek a symbolů

SD	–	Secure Digital - záznamové médium SD karta
MMC	–	Multi Media Card - záznamové médium
MEI	–	Matsushita Electric Company - společnost, která původně navrhla SD kartu
SDA	–	Secure Digital Association - společnost, která v současnosti zastřešuje specifikace SD karet
ECC	–	Error Correction Code - algoritmus pro opravu chyb
CSD	–	Card Specific Data - registr obsahující konfigurační informace pro přístup k datům (SD karta)
CID	–	Card Identification Data - registr obsahující informace, jednoznačně definující SD kartu (její ID, jméno, ...
USB	–	Universal Serial Bus - univerzální sériové rozhraní
FAT	–	File Alocation Table - tabulka souborového systému
EIA	–	Electronic Industries Association - společnost, jenž vyvinula rozhraní RS232
SPI	–	Serial Peripheral Interface - sériové rozhraní pro periferie
MCU	–	MicroControler Unit - mikropočítačová jednotka
RISC	–	Reduced Instruction Set Computer - redukovaná instrukční sada
CISC	–	Complex Instruction Set Computer - kompletní instrukční sada
RAM	–	Random Acces Memory - paměť používaná v počítačích a dalších elektronických přístrojích
NTFS	–	New Technology File System - souborový systém používaný moderními operačními systémy
MPT	–	Master Partition Table - hlavní tabulka oddílů
MBC	–	Master Boot Code - program pro spouštění bootovací sekvence
ANSI	–	American National Standarts Institute - americký institut
GNU	–	sada software pro operační systémy
GCC	–	GNU Compiler Collection - sada kompilátorů vytvořených v rámci projektu GNU
USART	–	Universal Serial Asynchronous Reciever and Transmitter - univerzální sériový, asynchronní přijímač a vysílač
LED	–	Light Emitting Diode - dioda emitující světlo

Obsah

1 Úvod

Sériové rozhraní RS232 bylo původně určeno k propojení počítače s modemem, ale postupem času se jeho používání rozšířilo i do dalších oblastí počítačové techniky a elektroniky. Toto rozhraní umožňuje vzájemnou sériovou komunikaci dvou zařízení, tzn. že jednotlivé bity přenášených dat jsou vysílány postupně za sebou po jediném vodiči. V současné době používání RS232 v oblasti osobních počítačů již téměř vymizelo a bylo nahrazeno výkonnějším sériovým rozhraním USB. Nicméně v průmyslových aplikacích je tento standart stále používán, hlavně jeho varianty RS422 a RS485. Jsou zde používány pro svou jednoduchost, protože pouze definují jak přenést určitou sekvenci bitů a nezabývají se vyššími vrstvami komunikace.

V dnešní době je velký počet průmyslových procesů řízen počítači. Nároky na objem a komplexnost dat neustále stoupají a ne vždy je vhodné používat klasický či průmyslový počítač s dostatečnou záznamovou kapacitou. Integrace těchto počítačů by byla nákladná a komplikovaná. Exituje celá řada případů kdy stačí používat jednoduché automaty, at'už pro řízení procesů nebo pro záznam dat, i když pro tyto účely nebyly původně navrženy. Například pro zaznamenávání dat je mnohem jednodušší doplnit systém o jednoúčelové zařízení pro záznam, které je k systému připojeno přes standartizované rozhraní.

Základním požadavkem tohoto zařízení je dostatečná záznamová kapacita a jednoduchá přenostitelnost zaznamenaných dat pro další zpracování. V tomto projektu jsem zvolil jako záznamové médium paměťovou kartu SD, protože její vlastnosti jsou pro tento účel více než dostatečné:

- dostatečná kapacita v rozsahu 16MB až 12GB a stále roste
- malé rozměry a spotřeba
- vysoká odolnost vůči mechanickému opotřebení
- kompaktibilita s PC, popř. PDA
- dostupnost
- cenově výhodná

Existují samozřejmě i jiné alternativy použití v podobě další typů paměťových karet. Jsou to např. paměťové karty typu MMC, CF a USB FLash disky. Poslední jmenované jsou omezeny potřebou UBS portu.

Pro snadnou přenositelnost zaznamenaných dat na PC je zapotřebí zvolit souborový systém pro paměťovou kartu, běžně podporovaný v současných operačních systémech. Pro tento účel jsem zvolil souborový systém FAT32, jenž je stále hodně používán nejen na PC a zároveň je jednoduchý pro implementaci na jednočipovém mikropočítači.

V následujícím textu jsou popsány principy činnosti sériového rozhraní RS232, dále pak paměťové karty SD. Další části jsou věnovány použitému hardwaru (MCU ATmega32, MAX232) a somozřejmě také použitému souborovému systému a software. Poté následuje část realizace, kde je popsán konkrétní návrh.

2 Rozhraní RS232

V 60. letech společnost známá jako EIA (Electronic Industries Association) vyvinula standartní rozhraní pro datovou komunikaci. V té době se data nejčastěji přenášela mezi počítači, z nichž jeden byl mainframe a druhý terminál (popř. mezi dvěma terminály), pomocí telefonních linek. K ukutečnění tohoto přenosu byly zapotřebí modemy na obou komunikujících stranách. Tento koncept však vykazoval chyby. Přenášena data byla často poškozena během přenosu po analogových linkách, a proto musel být proces přenosu opakován. Bylo zapotřeb vyvinout spolehlivý způsob propojení, který by byl navíc umožňoval komunikaci mezi zařízeními různých výrobců. Toto by navíc zjednodušilo výrobu hardwarového, ale i softwarového vybavení. Reakcí společnosti EIA na tyto požadavky bylo vytvoření rozhraní RS232.

Od doby kdy bylo rozhraní RS232 vytvořeno byly vyvinuty jeho varianty, jako např. EIA232F, RS422, RS485.

2.1 Popis elektrického rozhraní

2.1.1 Napěťové úrovně

RS232 používá dvě napěťové úrovně. Logickou 1 a 0. Log. 1 je někdy označována jako marking space nebo také klidový stav, log. 0 se označuje jako space state. Log. 1 je indikována zápornou napěťovou úrovní a log. 0 je indikována kladnou úrovní. Nejběžněji se pro generování napětí používá napěťový zdvojovač z 5V a invertor. Logické úrovně jsou potom přenášeny napětím +10V pro log. 0 a -10V pro log. 1.

2.1.2 Délka vedení

Standard pro RS232 uvádí jako maximální možnou délku vodičů 15m nebo délku vodiče o kapacitě 2500pF. To znamená, že při použití kvalitních vodičů lze dodržet standard a při zachování jmenovité kapacity prodloužit vzdálenost až na cca 50m.

Kabel lze také prodlužovat snížení přenosové rychlosti, protože potom bude přenos odolnější vůči velké kapacitě vedení. Uvedené parametry počítají s přenosovou rychlostí 1900Bd.

Texas Instruments uvádí tyto hodnoty pouze jako orientační, jsou totiž výsledkem laboratorních měření závislostí délek vodičů na přenosové rychlosti. V praxi je zapotřebí počítat s rušením a ostatními faktory ovlivňující přenos.

2.2 Popis fyzického rozhraní

2.2.1 Rozložení pinů

viz. obrázek 1

2.2.2 Význam pinů

Význam jednotlivých pinů je popsán v tabulce 1

Číslo pinu	Zkratka	Význam pinu	Popis
1	CD	Carrier Detection	detekce nosné, modem oznamuje terminálu, že na lince detekoval nosný signál
2	RXD	Recieve Data	tok dat z modemu do terminálu
3	TXD	Transmit Data	tok dat z terminálu do modemu
4	DTR	Data Terminal Ready	terminál oznamuje modemu, že je připraven komunikovat
5	DSR	Data Set Ready	modem oznamuje terminálu, že je připraven komunikovat
6	SGND	Signal Ground	signálová zem
7	RTS	Ready To Send	terminál oznamuje modemu, že komunikační cesta je volná
8	CTS	Clear To Send	modem oznamuje terminálu, že komunikační cesta je volná
9	RI	Ring Indicator	modem oznamuje terminálu, že na lince detekoval signál zvoní

Tabulka 1: Význam jednotlivých pinů

2.3 Přenosový rámec

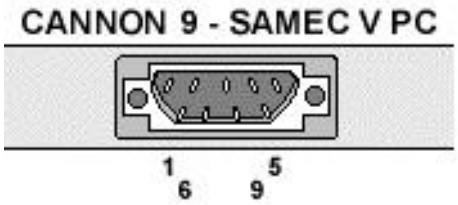
Kompletní přenosová skupina je tvořena přenášenými daty (7/8bitová) doplněná o START BIT, STOP BIT a PARITU. Přenosový rámec je tedy minimální přenášená skupina dat.

2.4 Datový přenos

RS232 používá asynchronní přenos dat konstantní přenosovou rychlostí. Každý přenesený byte je proto třeba synchronizovat. K synchronizaci se používá sestupná hrana signálu startovacího bitu, za ní se posílají data.

2.4.1 Syncronní přenos

Synchronní přenos informací znamená, že na nějakém vodiči nebo vodičích se nastaví určitá úroveň, která přenáší informací a validita se potvrzuje impulsem nebo změnou úrovně synchronizačního signálu. Synchronizačním signálem se tedy přenášené informace kvantují. Kvantování je diskretizace hodnot signálu (přiřazení hodnoty signálu v určitém časovém okamžiku dané diskrétní úrovni).



Obrázek 1: Rozložení pinů - samec

Vlastnosti

- Vhodná pro velké objemy informací, přenášených po více vodičích
- Nutně jednoznačně určit, kdo vysílá synchronizační implusy
- Možno použít spojitě proměnnou rychlost přenosu, například podle chybovosti
- Nutnost použití synchronizačního vodiče - je zde navíc, nepřenáší žádnou informací
- Přijímací ani vysílací zařízení nepotřebují nijak složitou elektroniku

2.4.2 Asyncronní přenos

Přenos informací probíhá v určitých sekvencích. Informace jsou přenášeny přesně danou přenosovou rychlostí a uvozeny startovací sekvencí, na kterou se synchronizují všechna přijímací zařízení. Všechny strany musí obsahovat přesný oscilátor, díky kterému se data posílají v přesně definovaných intervalech. Po ukončení přenosu sekvence je další příjem synchronizován další startovací sekvencí.

Vlastnosti

- Nevýhodný pro velké objemy dat, vhodný pro dlouhá vedení, na nichž by synchronizační vodič činil nezanedbatelné finanční náklady
- Lze použít pro komunikaci mezi mnoha zařízeními
- Nutno jednoznačně definovat přenosové rychlosti, každou změnu je nutné ošetřit softwarovou synchronizační sekvencí, jenž obsahuje příkaz, který hardwarově změní přenosovou rychlost

- Složitá a drahá elektronika pro přenos, nutno použít krystalové oscilátory
- Až o 20% menší přenosová rychlost užitečných dat při stejné rychlosti komunikace, vzhledem k nutnosti startovacích, paritních a dalších bitů

2.4.3 Parita

Parita je nejjednodušší způsob jak bez nároků na výpočetní výkon zabezpečit přenos dat. Ve vysílacím zařízení se sečte počet jedničkových bitů a přenášený rámec se doplní o paritní bit, jehož hodnota se nastaví tak, aby byla zachována předem dohodnutá podmínka sudého nebo lichého počtu jedničkových bitů.

- SUDÁ PARITA - počet jedničkových bitů + paritní bit = sudé číslo.
- LICHÁ PARITA - počet jedničkových bitů + paritní bit = liché číslo.
- SPACE PARITA - tzv. nulová parita, paritní bit je vždy v log. 0, používá se například při komunikaci 7bitového zařízení s 8bitovým, kdy paritní bit nahrazuje pevně nastavenou log. 0, jenž je posledním bitem v byte. Tím je zachována kompaktibilita s 8bitovým přenosem.
- MARK PARITA - paritní bit je nastaven na log. 1, při kompenzaci 7bitového provozu je třeba jej na přijímací straně nulovat, jinak není kompaktibilní s ASCII.

2.4.4 STAR / STOP bity

- START bit - počáteční bit, jehož hodnota musí být opačná než je klidová úroveň.
- STOP bit - definuje ukončení přenosu, zároveň zajišťuje určitou prodlevu pro přijímač, právě v době příjmu stop bitu většina zařízení zpracovává přijatý Byte.
- ZDVOJENÝ STOP bit - používá se u pomalejších zařízení pro doběh zpracování přijatého znaku, jedná se o standart na 110Bd.

2.4.5 Řízení toku dat - handshaking

Představuje potvrzení příjmu dat či připravenost k přenosu a jeho zahájení na úrovni hardwarového nebo softwarového rozhraní.

Hardwarový handshakinkg spočívá v přenosu vysílače k přijímači, vysílač tím oznamuje, že má data připravena k přenosu. Opačný přenos (od přijímače k vysílači) používá přijímač k oznámení, že je schopen data zpracovávat. Softwarový handshaking probíhá na úrovni komunikačních protokolů (ZMODEM, KERMIT, ...), pomocí běžného datového kanálu přijímač vysílači sdělí, zda je schopen data přijímat a zpracovávat. Například DOS / BIOS v počítačích PC používá pro SW handshaking znaky v ASCII tabulce (XON, XOF). Je-li však potřeba v toku dat tyto znaky vyslat, je nutné vyslat speciální sekvenci znaků, což přenos dat obsahující převážně tyto znaky značně zpomaluje.

3 Paměťová karta SD

SD karta (Secure Digital) je paměť založená na technologii Flash. Je navržena tak, aby poskytovala bezpečnost, potřebnou kapacitu, výkon a požadavky nových spotřebitelských zařízení, pracujících s např. audio a video soubory. Je rychlejší a schopna vyšší paměťové kapacity. Tento paměťový modul má v sobě zakomponován ochranný mechanismus podle bezpečnostních standartů SDMI. Podporuje vzájemnou autorizaci a obsahuje šifrovací algoritmus pro ochranu před ilegálním používáním obsahu paměti.

Komunikace je založena na 9ti pinovém rozhraní (clock, řízení, 4x data, 3x napájení). Operuje v nízkých napěťových úrovních. Toto rozhraní podporuje také standartní operace MMC (Multi Media Card), je tedy zachována kompaktibilita. Hlavní rozdíl mezi kartou SD a MMC je v inicializačním procesu. Původně byla SD karta navrhnuta společnostmi MEI (Matsushita Electric Company), Toshiba Corporation a SanDisk Corporation. V současnosti jsou specifikace kontrolovány společnosti Secure Digital Association (SDA).

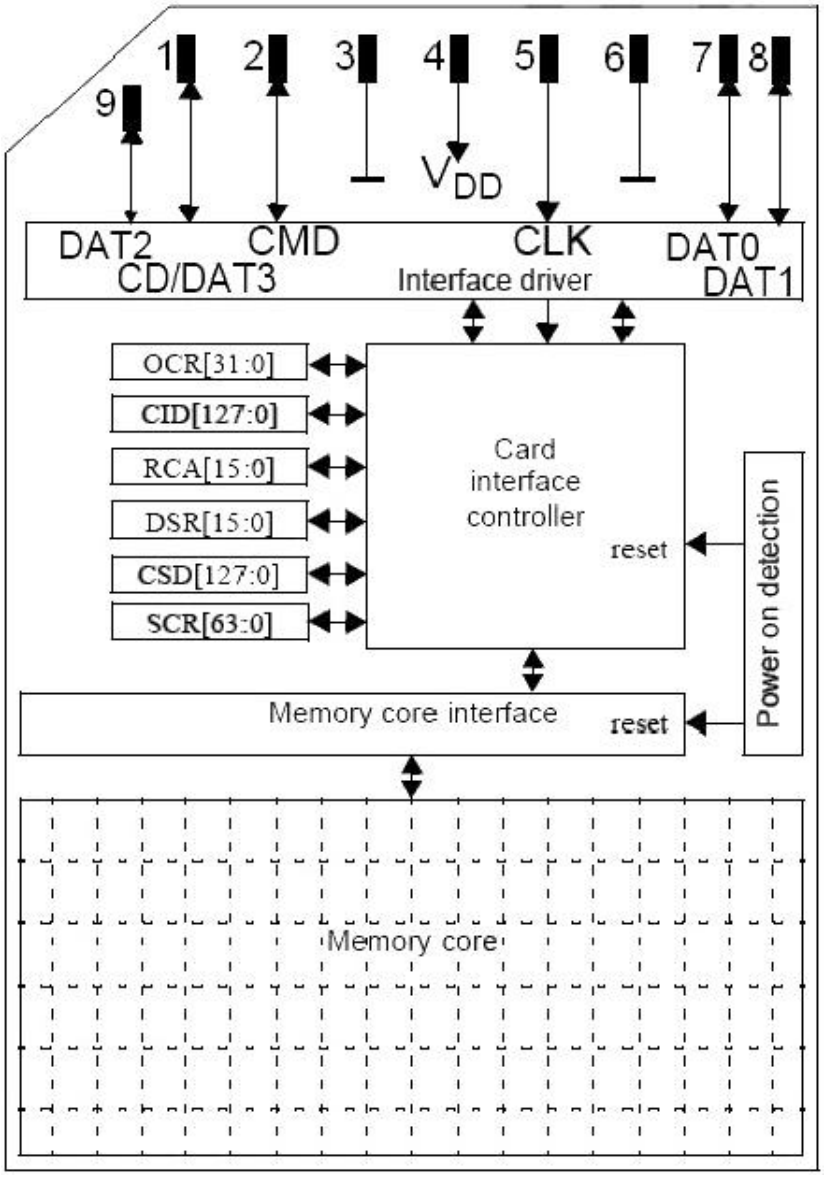
Rozhraní SD karty dovoluje snadnou integraci do jakékoliv technologie, bez ohledu na to, jaký používáme mikroprocesor. Současné SD karty nabízejí několika GB kapacitu paměti, je tedy vhodná pro uchovávání velkého objemu dat. Navíc nabízí možnost použití inteligentního řadiče pro řízení protokolů rozhraní (vestavěn přímo do karty), používání algoritmů pro copyright ochranu a pro opravu chyb ECC (Error Correction Code). Dále používá algoritmy určování defektů v datech, získávání ztracených dat zpět řízení napájení a hodinového signálu.

3.1 Technologie Flash

Paměti typu Flash jsou odvozeny od pamětí EEPROM a kombinují výhody několika dalších typů pamětí. Nejvýznamější z nich trvalé uložení dat (po odpojení napájení nezpůsobí ztrátu dat jako např. u pamětí typu RAM). Paměti Flash mají nízké napájecí napětí, malou spotřebu a jsou odolné vůči vlivům okolního prostředí. Další výhodou je velmi rychlá odezva na čtení a zápis.

3.2 Struktura SD - Fyzické rozhraní

Obal SD karty tvoří plastové pouzdro, na jedné straně je vyvedeno 9 kontaktů. Má rozměry 32 x 24m s tolerancí 0,1mm, tlouštka je 2,1mm (s tolerancí 0,15mm). Váha celé SD karty je maximálně 2g. Uživatel pro připojení k SD kartě používá 9ti pinový konektor. Obrázek 2 znázorňuje vnitřní strukturu SD karty. Je zde mimo jiné popsáno uspořádání konektorů v pouzdře, umístění hlavní řídící jednotky a také paměťové jádro. Důležitou funkci plní registry OCR, CID, CSD a SCR, které nesou specifické informace o kartě. Registry RCA, DSR jsou konfigurační registry, ve kterých jsou uloženy aktuální konfigurační informace. Detailně jsou popsány v tabulce 3. Poté následuje popis jednotlivých konktorů, je znázorněn v tabulce 2.



Obrázek 2: Pouzdro SD karty + vnitřní struktura

Číslo kontaktu	Zkratka	Význam kontaktu
1	CD/DAT3	Detekce karty/Datová linka, bit 3
2	CMD	Řízení karty
3	VSS1	Uzemnění
4	VDD	Napájení
6	VSS2	Uzemnění
7	DAT0	Datová linka, bit 0
8	DAT1	Datová linka, bit 1
9	DAT2	Datová linka, bit 2

Tabulka 2: Význam jednotlivých kontaktů

Název	Délka	Popis
CID	128	Card ID - obsahuje identifikační informace: jméno výrobku, id výrobce, datum výroby, a další
RCA	16	Relative Card Adress - nastavení adresy kary, nedostupný s SPI módu
DSR	16	Drive Stage Register
CSD	128	Card Specific Data - obsahuje informace ohledně přístupu do obsahu karty, např: formát dat
SCR	64	SD Configuration Register - registr nastavený při výrobě
OCR	32	Operation Conditional Register - ukládá VDD napětí karty, informace ohledně kapacity
SSR	512	SD Status Register - stavový registr
CSR	32	Card Status Register - info o stavu karty

Tabulka 3: Popis vnitřních registrů SD karty

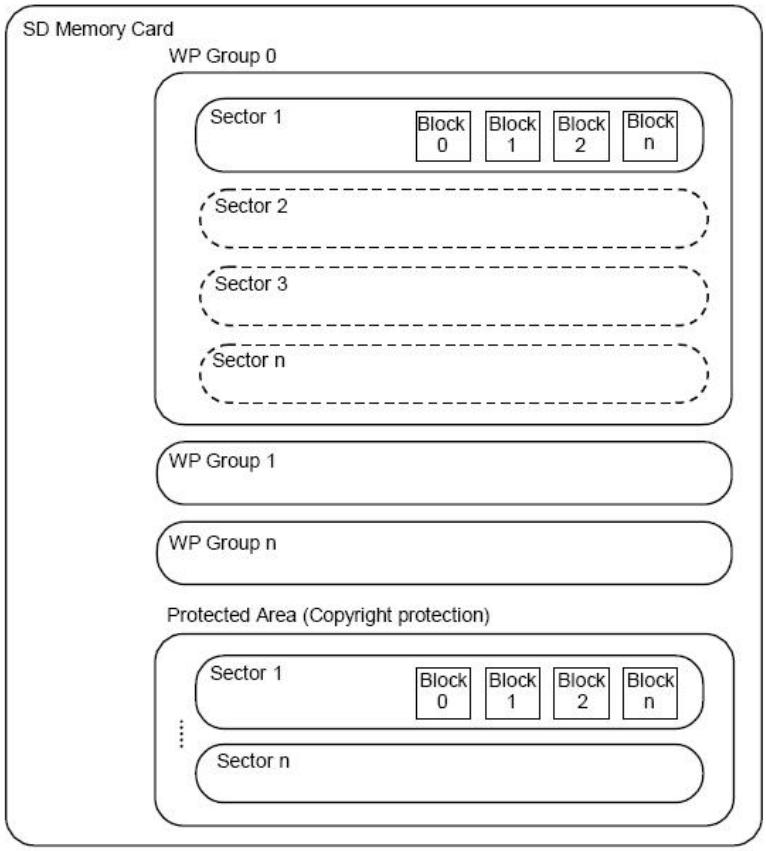
3.2.1 Organizace paměti

Základní datovou jednotkou přenášenou z/do SD karty je jeden Byte. Všechny operace přenášející data po blocích a potřebují znát délku bloku definují jeho délku jako celé číslo (toto číslo je násobek bajtu). Pro operace nepracující tímto způsobem musíme použít další rozdělení (viz obr. 3). Pro rozdělování paměti používáme tři základní pojmy:

Důležitou součástí pro organizaci paměti je CSD registr (Card Specific Data). Jsou v něm uloženy všechny potřebné informace o dané SD kartě a také různé konfigurace. Dále obsahuje informace o výrobci, data pro uživatele, ochranu proti zápisu/čtení a informace o používaných formátech. Uživatel může číst informace uložené v CSD a také upravovat jeho obsah (pouze data určená pro uživatele) pomocí příkazů SEND_CSD a PROGRAM_CSD.

Vlastnosti

- Blok - Tato jednotka souvisí s blokově orientovanými příkazy pro čtení a zápis, její velikost je počet bajtů, které byly přeneseny příkazem pro přenos bloku dat. Tato velikost je pevně stanovená, ale lze ji přeprogramováním změnit. Informace o povolené délce bloku je uložena v CSD.
- Sektor - Jednotka je spojena s příkazy pro mazání (nejsou blokově orientované). Její velikost je počet bloků mazaných jako jedna část. Velikost bloku je pevně stanovená pro každé zařízení zvlášť, je uložena v CSD.
- WP Skupina - Je minimální jednotka pro příkazy prováděné na zařízeních, které zahrnují ochranu proti zápisu. Její velikost je počet skupin, které budou chráněny proti zápisu. Velikost skupiny je stanovena pro každé zařízení zvlášť, je uložena v CSD.



Obrázek 3: Organizace paměti

3.2.2 Komunikace s SD kartou

Sběrnice SD karty obsahuje 6 komunikačních linek a 3 podpůrné (řídící), podle toho, jak jsou využívány se rozlišují 3 komunikační protokoly:

- Jednobitový SD mód
- Čtyřbitový SD mód
- SPI mód

Význam jednotlivých kontaktů v jednotlivých módech je popsán v tabulce 4.

Během inicializačního procesu jsou příkazy posílány do každé karty zvlášť. To dovoluje aplikaci pracující s kartou tuto kartu detekovat a přiřadit ji logikou adresu na fyzických

slotech. Data jsou posílána také do každé karty zvlášť. Avšak pro zjednodušení řízení zásobníku u SD karty, mohou být všechny příkazy (po inicializaci) posílány současně do všech karet připojených k zařízení. Adresování je prováděno pomocí informačního packetu.

Sběrnice SD karty dovoluje dynamickou konfiguraci očíslování datových linek. Po uvedení karty do provozu se pro přenos dat používá pouze linka DAT0. Po inicializaci může host měnit šířku sběrnice (počet aktivních datových linek). Tato vlastnost umožňuje doladit hardwarové požadavky s nároky na výkon celého systému.

3.2.3 SD protokol

Při použití SD protokolu jsou odděleny datové linky od řídících. Linka CMD slouží pro obousměrné zasílání příkazů a linka DAT (popř. DAT0 - DAT3) je pro přenos dat, také obousměrný. Linka CLK je pro vysílání hodinového signálu od hosta ke kartě.

3.2.4 SPI protokol

Při použití protokolu SPI jsou pro komunikaci využívány 3 základní komunikační vodiče a navíc signál CS (Chip Select). Signál CS se používá pro výběr zařízení pro komunikaci a musí být aktivní po celou dobu komunikace. Na rozdíl od SD protokolu jsou zde používány pouze linky DI a DO pro přenos řídících příkazů i dat.

V tomto zařízení je využita pouze jedna paměťová karta, tedy aktivace pomocí signálu CS je nejjednodušší způsob jak s kartou komunikovat. CRC je v tomto případě zbytečná a její výpočet by způsoboval nežádoucí prodlevu. Způsob zápisu a čtení dat po jednom bloku je zde naprosto dostačující. SPI protokol je tedy vhodný pro komunikaci SD karty s AVR.

Podstatou komunikace s SD kartou je odesílání příkazů a následovný příjem nebo vysílání dat. Datový proud začíná start bitem a končí stop bitem. Komunikace tedy ve třech krocích. Nejdříve je vyslán příkaz (command), tedy posloupnost znaků, která zahjuje přenos. Je posílán buď jako adresní příkaz (ze zařízení, které je host) nebo všem kartám ve formě broadcastu. V dalším kroku je poslána odpověď (response). Je to sada znaků odesílána z adresované karty nebo ze všech připojených karet jako odpověď na broadcast. V poslední etapě jsou zasílána požadovaná data z karty na host nebo obráceně.

3.2.5 SPI mód

SPI je volitelným módem, který má SD karta k dispozici. Je koncipován pro komunikaci po SPI sběrnici, kterou je vybavena většina mikrokontrolérů. Typ módu lze po připojení vybrat jen jednou, a to během prvního reset příkazu. Změna je možná pouze po odpojení a opětovném připojení karty ke sběrnici. V SPI módu neexistují žádné broadcast příkazy, výběr karty se provádí pomocí signálu CS.

Registry v SPI módu

V SPI módu jsou dostupné pouze registry OCR, CID a CSD. Jejich popis je uveden v tabulce 3.

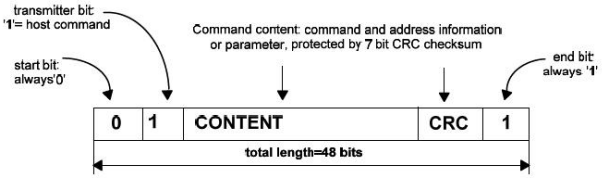
Komunikace v SPI módu

Komunikační protokol SPI je rozdělen na 3 části:

- command - příkaz
- response - odpověď
- data block - datový blok

Host připojený ke kartě kontroluje celou komunikaci. Vybraná karta reaguje na přijatý příkaz (command token) zasláním odpovědi ve formě response tokenu. Na každý přijatý datový blok reaguje zasláním response data tokenu.

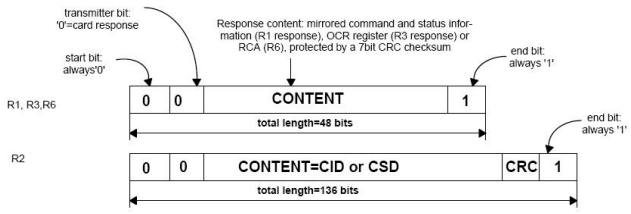
3.2.6 Příkazy v SPI módu



Obrázek 4: Struktura příkazu - Command

Každý příkaz předchází start bit (log. 0) a je ukončen stop bitem (log. 1). Má délku 48b a obsahuje start bit, stop bit, identifikace příkazu(1b), CRC (7b), dále pak číslo příkazu(6b) a argument(32b). Struktura příkazu je znázorněna na obrázku 4. Odpověď je dlouhá 48 nebo 136 bitů, to závisí na kontextu. Jeho struktura je obdobná, identifikace příkazu je nastavena v log. 0 (jedná se o komunikaci z karty na host). Odpověď je znázorněna na obrázku 5.

3.2.7 Odpověďi v SPI módu



Obrázek 5: Struktura odpovědi - Response

Odpověď typu R1

Karta zasílá tuto odpověď po každém přijatém příkazu kromě SEND_STATUS. Délka je 1B a nejvyšší bit (sedmý bit) je vždy v log. 0. Ostatní bity signalizují jednotlivé chyby.

Odpověď typu R1b

Tento typ odpovědi má stejnou strukturu, kromě signálu BUSY, který je volitený. Tento signál pokud je v log. 0 určuje, že karta je zaneprázdněna. V opačném případě je karta schopna přijímat další příkazy.

Odpověď typu R2

Karta zasílá tuto odpověď při příjmutí příkazu SEND_STATUS. Tato odpověď je 2B dlouhá, první Byte je shodný s odpověďí typu R1. Obsah druhého Bytu signalizuje různé stavy karty, např. karta je zamčena nebo má aktivní ochranu proti zápisu.

Odpověď typu R3

Karta zasílá tuto odpověď při příjmu příkazu READ_OCR. Odpověď R3 je 5B dlouhá, první Byte je shodný s odpovědí R1, zbytek tvoří obsah registru OCR.

Data response - odpověď na datový blok

Každý datový blok zapsaný na kartu je potvrzen pomocí odpověďi data response. Tento token je 1B dlouhý a obsahuje informace o správnosti příjmu a zápisu dat, případně o chybách.

3.2.8 Datové tokeny v SPI módu

Příkazy pro čtení a zápis jsou spřaženy s datovými tokeny, přes které jsou data přijímána a vysílána. Datové tokeny mají následující struktru.

- 1. Byte - příznak začátku datového bloku
- Uživatelská data
- poslední 2. Byty - CRC

3.3 Elektrické rozhraní

3.3.1 Uvedení do provozu - inicializace

Po připojení SD karty ke sběrnici a k napájení je karty v SD módu. Pro přechod do SPI módu je nutné:

- převést signál CS na neaktivní úroveň
- vyčkat 74 hodinových cyklů sběrnice SPI
- převést signál CS na aktivní úroveň
- zaslat kartě příkaz CMD0 - přechod do SPI módu
- vyčkat na odpověď typu R1, návratová hodnota musí být 0x01, jinak indikuje chybu
- zaslat kartě příkaz ACMD41 - pro rozpoznání zda se jedná o MMC nebo SD
- vyčkat na odpověď R1 - pokud odpověď příjde, může začít přenos (pokud ne, jedná se o MMC)

Po uvedení do provozu, včetně tzv. “hot insertion ”(vložení SD karty v době, kdy sběrnice pracuje) SD karta vstupuje do klidového stavu. Během trvání tohoto stavu SD karta ignoruje všechny ostatní příkazy, přicházející ze sběrnice, dokud neobdrží příkaz ACMD41 (typ příkazu ACMD by měl vždy předcházet příkazu CMD55).

ACMD41 je speciální synchronizační příkaz pro určení rozmezí napájecího napětí a pro průzkum SD karty před ukončení sekvence uvedení do provozu. Provádění příkazu ACMD41 je určeno speciální značkou, která určuje, že karta stále provádí sekvenci uvedení do provozu, a že ještě není připravena pro indentifikaci. Tento bit je znamením pro obvod, který bude s SD kartou komunikovat (host). Tento obvod musí počkat dokud karta nedokončí sekvenci uvedení do provozu, pravidelně kontroluje připravenost karty ke komunikaci. Maximální doba power up sekvence by neměla překročit 1 sekundu.

Po úspěšném provedení power up sekvence začíná host vysílat hodinový signál a začíná s inicializační sekvencí, kterou posílá přes CMD linku. Je to sekvence log. 1. Maximální doba této sekvence je 1 ms, tedy 74 hodinových implulsů. Poté za dalších 64 hodinových impulsů je karta připravena ke komunikaci. Toto všechno se provádí k eliminaci problémů souvisejících s uvedením karty do provozu.

Každý master na sběrnici by měl mít implementovány příkazy ACMD41 a CMD1. CMD1 je určen k dotazování multimediálních karet (MMC), aby masterovi zaslaly jejich stav při provozu.

Inicializační proces implicitně deaktivuje provádění kontroly pomocí CRC, ale při inicializaci je nutné k prvnímu zaslanému příkazu (CMD0) přiřadit platný kontrolní součet. Ten je předem známý a je uveden ve specifikacích. Má hodnotu 0x95.

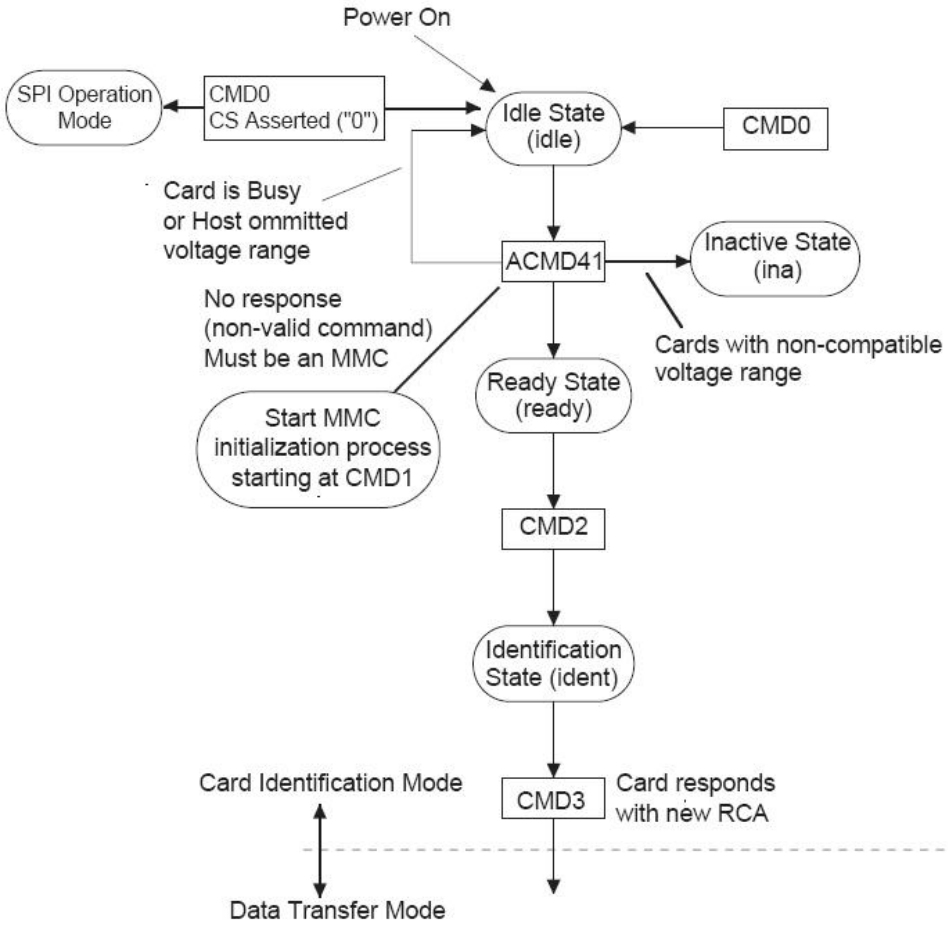
Po připojení karty k napájení je karta v SD módu, je tedy nutné ji přepnout do módu SPI. Na obrázku 6 je uveden postup inicializace SD karty v SPI módu.

3.3.2 Specifikace SD karty

Následující hodnoty jsou definovány pro nominální teploty okolí a velikosti napájecího napětí.

Specifikace okolních podmínek

SD karta je konstruována pro rozmezí teplot od -25°C do 85°C, při vlhkosti vzduchu pohybující se v rozmezí od 25% do 95%. Obsahuje rovněž tzv. ESD protection (Electrostatic Discharge). Je to technologie, která zajišťuje ochranu před napěťovými pulzy z připojených zařízení či z okolí.



Obrázek 6: Inicializace SPI módu SD karty

Napájecí úrovně

Hodnoty napájecího napětí se pohybují v rozmezí 2,7V - 3,6V. Povolená hodnota zvlnění napájecího napětí je 60mV (špička-špička). Hodnoty proudů pro jednotlivé stavy SD karty (klidový stav, čtení, zápis jsou uvedeny v tabulce 7.

Číslo pinu	4bitový SD mód		1bitový SD mód		SPI mód	
1	CD/DAT3	Datová linka 3	N/A	není použit	CS	Vybrání karty
2	CMD	Zadávání příkazů	CMD	Zadávání příkazů	DI	Datový vstup
3	VSS1	GND	VSS1	GND	VSS1	GND
4	VDD	Napájení	VDD	Napájení	VDD	Napájení
5	CLK	Clock	CLK	Clock	CLK	Clock
6	VSS2	GND	VSS2	GND	VSS2	GND
7	DAT0	Datová linka 0	DAT0	Datová linka 0	DO	Datový výstup
8	DAT1	Datová linka 1 nebo přerušení	IRQ	Přerušení	IRQ	Přerušení
9	DAT2	Datová linka 2 nebo čekání na čtení	RW	Čekání na čtení	N/A	není použito

Tabulka 4: Význam jednotlivý kontaktů v různých komunikačních módech

Číslo příkazu	Argument	Odpověď	Popis
CMD0	nic	R1	softwarový reset
CMD9	nic	R1	zaslání CSD
CMD10	nic	R1	zaslání CID
CMD16	dél. bloku	R1	nastavení délky datového bloku
CMD17	adresa dat	R1	čtení dat z dané adresy
CMD24	adresa dat	R1	zápis dat na danou adresy
CMD58	nic	R3	čtení OCR
CMD59	0b	R1	nastavení CRC, podle bitu 0 u argumentu

Tabulka 5: Výpis některých příkazů

Bit	Chyba	Popis
0	Idle stav	Karta je ve stavu Idle a provádí inicializaci
1	Reset mazání	Sekvence mazání byla přerušena
2	Chybný příkaz	Byl přijat příkaz s neznámým kódem
3	Chyba CRC	Kontrolní součet u posledního příkazu selhal
4	Chyba mazání	Nastala chyba v sekvvenci mazání
5	Chyba adresy	Neplatná adresa
6	Chyba parametru	Parametr příkazu byl mimo rozsah

Tabulka 6: Význam bitů určující chybu u odpověďi typu R1

Stav	Hodnota	Jednotka
Klidový	250	uA
Čtení	65	mA
Zápis	75	mA

Tabulka 7: Hodnoty napájecího napětí

4 Hardwarové vybavení

4.1 Procesor Atmel AVR

Procesory typu AVR od firmy Atmel patří mezi tzv. MCU (MicroControler Unit) - jednočipový počítač je monolitický integrovaný obvod. Oproti klasickým procesorům mají MCU v čipu integrováno i několik dalších obvodů tak, aby byl MCU použitelný jako samostatný počítač. Obvykle se MCU skládá z těchto částí:

- CPU - od 4bitových až po 32bitové procesory
- Operační paměť - typu RAM, od několika Bytů až po KB
- Paměť programu a dat - ROM, EPROM, EEPROM, Flash obsahující instrukce procesoru a data
- Periferní obvody

Mnoho MCU je založeno na Hardvardské architektuře, mají tedy oddělené sběrnice instrukcí a dat, což jim na rozdíl od architektury von Neumann dovoluje souběžné načítání instrukcí a dat. To je pro CPU typu RISC výhodné, protože jejich hlavním znakem je dokončení jedné instrukce za jeden strojový cyklus. To znamená, že instrukce i data mohou být připravena při každém začátku strojového cyklu. To rovněž umožňuje mít datovou sběrnici o jiné bitové šířce než je šířka instrukční sběrnice.

Koncepce RISC (Reduced Instruction Set Computer) je založena na předpokladu, že frekvence používání složitých instrukcí je tak malá, že se nevyplatí je integrovat do čipu, a v případě potřeby jsou nahrazeny posloupnostmi jednoduchých instrukcí. Celá instrukční sada obsahuje malý počet jednoduchých instrukcí (cca 30). Díky tomu jsou v čipu jednodušší řídící obvody a zkracuje se doba zpracování instrukcí. Řídící obvody zabírají 6% až 10% a výkon instrukce s výjimkou komunikace s pamětí je jeden strojový cyklus. Protějškem této koncepce je CISC (Complex Instruction Set Computer), jenž má instrukční soubor s takovými instrukcemi, které pod jedním operačním kódem vykonají i složité operace s variabilitou různých adresovacích módů. Toto se provádí za cenu zpracování těchto instrukcí v několika strojových cyklech. Řídící obvody tvoří 60% plochy na čipu. Pro dosažení co nejvyšší rychlosti omezené přístupy do paměti se u koncepce RISC ušetřené místo na čipu využije pro soubory registrů, k nimž je jednocyklový přístup.

4.1.1 Popis procesoru Atmel AVR

Pro bakalářskou práci jsem použil procesor AVR ATmega. Tyto procesory mají 4 - 256KB paměti programu, 28 - 100pinová pouzdra a rozšířený instrukční soubor (instrukce násobení, dělení a instrukce pro lepší obsluhu většího množství paměti).

Všechny procesory AVR mají 32 8bitových registrů pro všeobecné použití, mohou obsahovat data i adresy. Posledních 6 registrů lze použít ve dvojici jako ukazatele adresy pro nepřímé adresování paměti dat. Tyto registry jsou označovány X, Y a Z. Procesory AVR mají 5 adresovacích módů pro paměť dat:

- přímé adresování
- nepřímé adresování
- nepřímé adresování s posunutím (6bitový posun)
- nepřímé adresování s inkrementací ukazatele adresy po zpracování instrukce
- nepřímé adresování s dekrementací ukazatele adresy před zpracováním instrukce

Přímé adresování spočívá v přímém zadání adresy do instrukce buď v číselné podobě nebo zástupným řetězcem reprezentujícím adresu daného registru. Oproti tomu nepřímé adresování spočívá ve využití tzv. ukazatelů. Ukazatel je 16bitové číslo (dvojice registrů X, Y, Z) obsahující adresu zdrojového nebo cílového bajtu. Ukazatel X pro adresu určenou pro registry R26 a R27, ukazatel Y pro R28 a R29, ukazatel Z pro R30 a R31. Každý z ukazatelů má své specifické použití. Pro nepřímé adresování a pro nepřímé adresování s post-inkrementem a pre-dekrementem se využívá všech tří ukazatelů. Pro nepřímé adresování s posunem jsou využity ukazatele X a Y. Nepřímé adresování lze využít i pro adresaci paměti programu, například pro načítání tabulek s daty reprezentující obrázek a jiné.

4.1.2 Paměť programu (Flash)

Instrukce programu jsou uloženy v nevolatilní (dlouhodobé) paměti Flash, která je integrována do pouzdra. Ačkoli jsou AVR 8bitové MCU, opcode každé instrukce má 16 bitů. Při klasifikování bitovosti se bere v úvahu nejdelší délka dat, kterou je AVR schopen zpracovat v jedné instrukci. Proto jsou AVR 8bitové procesory.

Flash je kvůli 16bitovým instrukcím adresována po 16 bitech. Často je využívána i pro ukládání konstantních dat, aby se ušetřila paměť RAM. Adresní prostor Flash je podle Hardvardské koncepce oddělen od adresního prostoru RAM, což způsobuje potíže s kompaktibilitou ukazatelů jazyka C, například při používání textových řetězců v paměti Flash.

4.1.3 Interní paměť dat

Datový adresní prostor se skládá ze souboru registrů, IO registů a SRAM. AVR mají 32 8bitových registrů a jsou klasifikovány jako RISC procesory.

Pracovní sada registrů je mapována na prvních 32 adres paměti (0000 - 001F) následovaných 64 IO registry (0020 - 005F). Skutečná SRAM začíná sekcemi registrů (od 0060). Na některých rozsáhlejších zařízeních může být prostor IO registrů větší, zabírají tedy větší část adresového prostoru SRAM.

4.1.4 Vykonávání programu

MCU rodiny AVR mají jednoúrovňovou pipeline, to znamená, že další strojová instrukce se načítá, když se aktuální instrukce vykonává. Zpracování většiny instrukcí trvá jeden strojový cyklus, AVR se tedy řadí mezi rychlé 8bitové MCU.

Rodina procesorů AVR byla navržena pro efektivní vykonávání kompilovaného kódu jazyka C. Paměť RAM u AVR tvoří souvislý celek, což má pozitivní vliv na jednoduchost výsledného programu. Další výhodou je 16bitový stack pointer. Díky tomu je možné ukazovat na kteroukoliv buňku v RAM a používat více lokálních proměnných.

4.1.5 USART

MCU AVR ATmega32 má dva programovatelné USARTy (Universal Synchronous/Asynchronous Reciever and Transmitter) pro sériovou komunikaci. Jeden z nich sdílí piny s programovacím rozhraním SPI, takže pro komunikaci s RS232 je použitelný jen jeden. Některé schopnosti USARTu:

- plný duplex (oddělené přijímací a vysílací registry)
- generátor přenosové rychlosti s jemným dělením
- podpora sériových rámců (5 - 9 datových bitů, 1 start-bit, 1 nebo 2 stop-bity)
- hardwarový generátor sudé i liché parity a kontrola parity
- šumové filtry (digitální LP filtr) a detekce falešného bitu
- 3 přerušení generovaná ukončením vysílání nebo příjmu a prázdným vysílacím registrem

USART se skládá ze tří základních celků - generátor přenosové rychlosti, vysílač a přijímač. Pro nastavení požadované přenosové rychlosti se používá 16bitový registr UBBR. Konkrétní rychlost podle hodnoty v UBBR (0 - 4095) a potřebnou hodnotu v UBBR pro požadovanou přenosovou rychlost zjistíme podle vztahu:

$$\mathbf{UBRR} = \frac{\mathbf{Fosc}}{16 \times \mathbf{BAUD}} - 1$$

kde: **UBRR** - hodnota v UBRR registru, **Fosc** - kmitočet procesoru, **BAUD** - bitová rychlost v Baudech

Vysílač obsahuje posuvný registr, do kterého se zápisem datového registru UDR vloží vysílaná data, která jsou dále zpracována. Jakmile je UDR připraven pro další zápis dat, je vyvoláno přerušení (je-li povoleno). Podle požadavku je vypočtena sudá nebo lichá parita (příp. žádná parita). Řídící logika pak podle rytmu hodinového signálu generátoru bitové rychlosti vysílá jednotlivé bity doplněné o start- a stop-bity a paritu přes výstupní budič na pin TxD. Jakmile je přenos dokončen, je vyvoláno další přerušení (je-li povoleno).

Přijímač přijímá sériový tok bitů přes pin RxD. Nejdříve se provádí filtrace za účelem odstranění šumů a rušení, poté regenerace tvaru signálu. Rekonstruovaným hodinovým signálem řídí procesor přijímací posuvný registr, který načítá přijaté datové bity. Z přijaté sekvence bitů se spočítá a zkontroluje parita. Po dokončení příjmu datového slova je vyvoláno přerušení (je-li povoleno) a přijatá data je možné přečíst z registru UDR.

Další nastavení USARTu a čtení stavových informací se provádí přes tři 8bitové registry UCSRA, UCSRB a UCSRC. Význam jednotlivých bitů v konfiguračním a stavovém registru UCSRA:

- RXC - je-li nastaven na log. 1, znamená, že registr UDR obsahuje přijatá data připravená k přečtení a může být vygenerováno přerušení
- TXC - po odeslání dat z posuvného registru je nastaven do log. 1 a může být vygenerováno přerušení
- UDRE - je-li nastaven na log. 1, lze do registru UDR zapisovat nová data k odeslání, může být generováno přerušení
- FE - v log. 1 signalizuje chybu rámce v přijatém slovu (které je v registru UDR), jeho přečtením se flag automaticky vynuluje
- DOR - v log. 1 signalizuje přetečení dat v registru UDR (data nebyla včas přečtena, posuvný registr je tudíž plný a přijímač detekoval další start-bit)
- UPE - v log. 1 signalizuje chybu parity přijatého slova v registru UDR
- U2X - nastavením do log. 1 se zvýší bitová rychlost na dvojnásobek (používá se při asynchronním přenosu), při synchronním přenosu je nastaven v log. 0
- MPCM - nastavením do log. 1 se aktivuje režim podpory víceprocesorové komunikace

Význam jednotlivých bitů v konfiguračním a stavovém registru UCSRB:

- RXCIE - v log. 1 je povoleno generování přerušení po dokončení příjmu (RXC je v log. 1)
- TXCIE - v log. 1 je povoleno generování přerušení po dokončení vysílání (TXC je v log. 1)

- UDRIE - v log. 1 je povoleno generování přerušení při vyprázdnění vysílacího registru UDR (UDRE je v log. 1)
- RXEN - nastavením do log. 1 se aktivuje blok přijímače
- TXEN - nastavením do log. 1 se aktivuje blok vysílače
- UCZS2 - slouží pro nastavení počtu datových bitů v rámci
- RXB8 - datový bit 8, pro příjem 9bitového slova
- TXB8 - datový bit 8, pro zápis 9bitového slova

Význam jednotlivých bitů v konfiguračním a stavovém registru UCSRC:

- UMSEL - nastavení synchronního (log. 1) a asynchronního (log. 0) režimu
- UPM0,UPM1 - nastavení parity (00- žádná parita, 10- sudá, 11- lichá)
- USBS - nastavení počtu stop-bitů (log. 0 pro jeden, log. 1 pro dva)
- UCPL - nastavení polarity hodinového signálu pro synchronní přenos - příjem na sestupnou hranu (log. 0), na vzestupnou hranu (log. 1), pro asynchronní přenos má být nastaven v log. 0

4.2 Základní blokový diagram zařízení

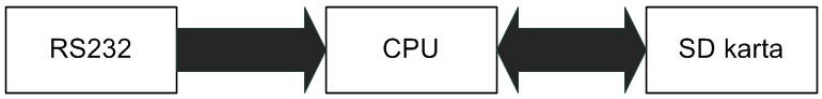
Blokové schéma je znázorněno na obrázku 7.

4.3 Převodník úrovní pro rozhraní RS232

RS232 je rozhraní pro sériový přenos informací, které bylo původně vytvořeno pro komunikaci dvou zařízení do vzdálenosti 20m. Pro větší odolnost proti rušení jsou informace po vodičích přenášeny větším napětím než je standardních 5V. Běžné PC jsou vybaveny jedním až dvěma porty RS232. U přenosných počítačů toto rozhraní již téměř vymizelo. Existují však převodníky RS232/USB (ve Windows se toto rozhraní chová jako virtuální port COM).

Protože USART u procesoru ATmega používá standardní 5V úrovně, musí se pro komunikaci s PC použít převodník úrovní na RS232. Úrovně jsou popsány v tabulce 8

Pro tento účel se často používá rozšířený a velice snadno dostupný integrovaný obvod MAX232, který obsahuje dva páry převodníků úrovní. Hlavní výhodou je, že ke své funkci potřebuje napájení jen 5V. Obsahuje totiž zdvojovače a invertory na principu nábojové pumpy, které vytvoří potřebná kladná a záporná napětí pro buzení RS232. Počet potřebných vnějších součástek je omezen na čtyři kondenzátory pro nábojové pumpy a jeden blokovací kondenzátor pro napájecí napětí. Schéma zapojení převodníku je součástí záznamníku.



Obrázek 7: Blokové schéma zařízení

Úroveň	**Na straně vysílače**	**Na straně přijímače**
log. 0	+5V až +15V	+3V až +25V
log. 1	-5V až -15V	-3V až -25V
Zakázaná	-3V až +3V	-3V až +3V

Tabulka 8: Napěťové úrovně RS232 pro vysílač a přijímač

5 Softwarové vybavení

5.1 Souborový systém

Postupným rozvojem počítačů a počítačových systémů bylo nutné ukládat, načítat a organizovat data. Aby tyto operace bylo vůbec možné provádět, byly vyvinuty souborové systémy, které definují, jak data (datové soubory) ukládat a indexovat v paměti (u PC je to nejčastěji harddisk). Struktura pevného disku je na obrázku 8. Druhy současně používaných systémů:

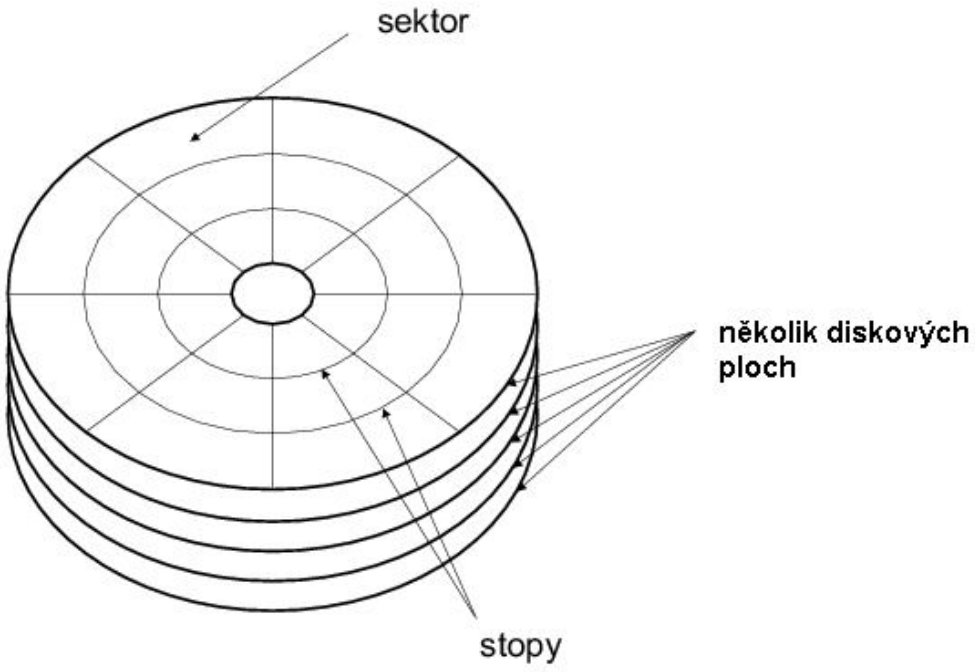
- FAT12, FAT16, FAT32 - File Allocation Table
- NTFS - New Technology File System
- Ext2, Ext3 - Extended Filesystem

Nejmodernější souborové systémy (NTFS, Ext3) se řadí mezi žurnálové souborové systémy, které zapisují změny do záznamu (journal). Jednotlivé operace jsou řazeny do transakcí. Operace se buď provedou v pořádku celé, nebo se neprovedou vůbec. Účelem této technologie je zvýšení ochrany dat před ztrátou integrity v důsledku náhlého přerušení dodávky elektrického proudu nebo pádu systému.

Nejmenší adresovatelnou jednotkou na disku je sektor, jenž má ve většině případů velikost 512 Bytů. Z několika sektorů se skládá alokační jednotka (cluster). Počet sektorů v clusteru je roven násobku n-té mocniny 2. Obrázek 8 ukazuje teoretické uspořádání sektorů na disku. Ve skutečnosti je na vnější části disku jiný počet sektorů ve stopě než na vnitřní. U současných disků není známo jejich reálné uspořádání sektorů; toto uspořádání není zveřejněno ani operačnímu systému. Proto mají tyto disky implementovanou dekódovací logiku pro určení pozice sektoru. Každý harddisk musí mít pevně stanovený bod, kde jsou uloženy všechny klíčové informace o disku, např. počet oddílů, jejich seřazení, používaný typ souborového systému apod. Na disku mohou být definovány až čtyři primární oddíly, z toho může být jeden rozšířený. Uspořádání jednotlivých oddílů a jejich umístění je definováno v MBR (Master Boot Record). MBR je umístěn v prvních 512 bytech na disku a obsahuje následující:

- MPT - Master Partition Table
- MBC - Master Boot Code

MPT je tabulka obsahující popisy oddílů nacházejících se na disku. Její velikost je omezena na informace o 4 oddílech, tzv. Primary Partitions. Jeden z těchto oddílů může být rozdělen na logické oddíly (Logical Partitions), kde je jeden z nich označen jako aktivní a počítač ho může používat k bootování.



Obrázek 8: Struktura pevného disku

MBC je malý počáteční bootovací program (bios load) a je obsažen v MBR. Tento program vybírá oddíl, který je určen k bootování. Ve většině případů obsahuje operační systém.

S takovým rozdělením disku je úzce spjato logické a fyzické adresování sektorů na disku. Logické adresování oddílů je počítáno od nuly, naopak při fyzickém adresování je číslování oddílu vedeno od začátku disku (MBR) a je inkrementováno přes celý disk. Toto rozdělení umožňuje používání více operačních a souborových systémů nebo je možné oddělit data od operačního systému.

Pro další rozdělení oddílů jsou určeny tzv. Partition Tables, jsou umístěny na 446. Bytu v MBR a jejich celková velikost je 64 Bytů. Tvoří 4 části po 16 Bytech (4 tabulky rozdělení oddílů), každá část popisuje jeden oddíl na disku. Struktura Partition Table je popsána v tabulce 9.

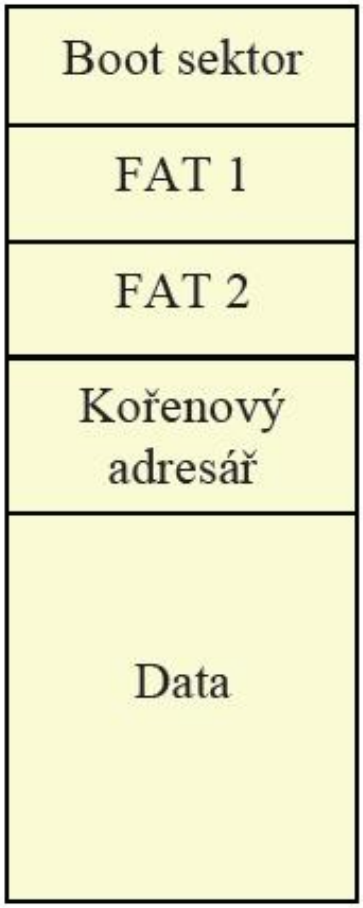
Offset	Velikost	Popis
0x00	1B	určuje stav (0x80 systémový, 0x00 nesystémový oddíl)
0x01	1B	začátek oddílu - hlava
0x02	2B	začátek oddílu - cylindr a sektor
0x04	1B	typ oddílu
0x05	1B	konec oddílu - hlava
0x06	2B	konec oddílu - cylind a sektor
0x08	4B	počet sektorů mezi MBR a prvním sektorem v oddílu
0x0c	4B	počet sektorů v oddílu

Tabulka 9: Partition Table v MBR

5.2 FAT32

Číslo v názvu soborového systému typu FAT udává velikost položky ve FAT tabulce. Základní principy a struktura jsou u všech verzí FAT v zásadě stejné, liší se jen v detailech. Na začátku oddílu je boot sektor obsahující základní informace o souborovém systému, který je v aktuálním oddílu nainstalován. Struktura boot sektoru je znázorněna v tabulce 10, jsou zde obsaženy základní parametry důležité pro práci se systémem FAT32.

Master Boot Record je uložen v prvním sektoru média, jeho adresa je 0x00000000. Dále pak obsahuje tabulku paměťového média v podobě čtyř 16-Bytových záznamů. Je tedy možné rozdělení na čtyři logické disky. Každý z těchto záznamů obsahuje informace o aktivním oddílu, logickou adresu prvního sektoru oddílu a velikost oddílu. V případě tohoto projektu je důležitý pouze první záznam.



Obrázek 9: Struktura FAT32

Nejprve je nutné načíst MBR a z prvního záznamu získat adresu začátku oddílu. Tímto způsobem je možné získat adresu boot sektoru a je možné jej načíst. Boot sektor je prvním sektorem oddílu a jsou v něm uloženy veškeré informace o nastavení oddílu.

Offset	Velikost	Popis
0x00	3B	skoková instrukce kódu
0x03	8B	jmenovka disku
0x0a	2B	počet bytů na sektor
0x0c	1B	počet sektorů na alokační jednotku (cluster)
0x0d	2B	počet rezervovaných sektorů před první FAT tabulkou od začátku oddílu
0x0f	1B	počet FAT tabulek, téměř vždy jsou 2
0x10	2B	tato část slouží pro FAT12 a FAT16, pro FAT32 musí být 0
0x1f	4B	množství sektorů v oddílu
0x24	4B	velikost jedné FAT tabulky v sektorech, specifické pro FAT32
0x2c	4B	cluster kde začíná root, obvykle 2 (není požadován), specifické pro FAT32

Tabulka 10: Položky v boot sektoru

Poslední 2 byty v boot sektoru musí mít hodnotu 0xaa55; pokud ne, souborový systém je poškozen. Parametry uvedené v tabulkách 9 a 10 se nastavují při formátování média, některé z nich jsou nastavovány v závislosti na velikosti média a typu ukládaných dat (např. velké či malé soubory). Při inicializaci je zapotřebí nejprve zjistit verzi souborového systému. Tu určíme podle celkového počtu alokačních jednotek (clusterů) nebo načteme boot sektor a ověříme, zda se jedná o FAT32.

Dalším krokem je zjistit adresy všech hlavních oblastí (adresa FAT tabulky, adresa začátku kořenového adresáře a adresa začátku datové oblasti), které jsou důležité pro obsluhu FAT:

- adresa_FAT ādresa_boot + (bytu_na_sektor x pocet_rezerv_bytu)
- adresa_root ādresa_boot + (bytu_na_sektor x (pocet_rezerv_bytu + (pocet_FAT x pocet_sek_na_FAT)))
- adresa_data ādresa_root + (max_polozek_root x 32)

Dalším elementem ve struktuře FAT32 jsou 2 FAT tabulky - primární a sekundární, přičemž sekundární je kopií primární. Pro každý cluster na disku je ve FAT přiřazen

32bitový záznam, jehož hodnota ukazuje na další cluster a určuje koncový nebo vadný cluster. Tento záznam je uložen ve formátu little endian.

Hodnota	Význam
0x?0000000	volný cluster
0x?0000001	rezervovaná hodnota, nepoužívá se
0x?0000002 - 0x?ffffef	použitý cluster, hodnota ukazuje na další cluster
0x?ffffff0 - 0x?ffffff6	rezervovaná hodnota, nepoužívá se
0x?ffffff7	rezervovaný nebo vadný cluster
0x?ffffff8 - 0x?fffffff	poslední cluster v souboru

Tabulka 11: Význam hodnot clusterů v tabulce FAT

5.2.1 Práce s root adresářem

Každý soubor nebo adresář je v kořenovém adresáři reprezentován jako 32B záznam. Tento záznam je detailně popsán v tabulce 12.

Offset	Délka	Popis
0x00	8B	Název položky
0x08	3B	Přípona položky
0x0B	1B	Příznak (souboru/adresáře) + atributy
0x1A	2B	První cluster souboru/adresáře
0x1C	4B	Délka položky

Tabulka 12: Struktura položky v kořenovém adresáři

Při zápisu nového souboru je nutné načíst sektor s adresou kořenového adresáře a podle toho najít volné místo pro nový záznam. Pokud je volné místo nalezeno je možné pokračovat se zápisem nového souboru. Tento postup se skládá:

- Na adresu volné položky je k jejímu názvu přiřazeno jméno nového souboru
- Zápis pozice prvního volného clusteru pro data.
- Zápis velikosti souboru (výpočet: 512B x pocet_sektoru_na_cluster)

Při zápisu nových dat do souboru se zvětšuje jeho velikost, proto je nutné podle množství zapsaných dat změnit i velikost položky v kořenovém adresáři.

5.2.2 Práce s FAT

FAT tabulka slouží pro mapování použitých clusterů v datové oblasti. Každá položka má ve FAT32 délku 4B a představuje jeden cluster v oblasti určené pro data. První dva

záznamy nemají žádný vztah k datové oblasti a jsou pevně stanovené. Záznamy za nimi již označují datové clustery.

Pokud položka ve FAT tabulce obsahuje hodnotu 0x00000000, poukazuje to na prázdný cluster a je možné jej použít pro zápis dat. Pokud však obsahuje hodnotu 0xFFFFFFFF, jedná se o konec souboru - na tomto clusteru soubor končí.

Položka obsahující hodnotu v rozmezí 0x00000002 - 0xFFFFFFEF určuje, že je na její pozici umístěn datový soubor. Často má datový soubor větší velikost než je velikost clusteru, a proto je často jeden soubor rozložen na několika clusterech. Hodnota v položce pak ukazuje na další cluster, na kterém soubor pokračuje.

5.3 Hlavní části programu

5.3.1 Inicializace SPI

SPI modul u mikrokontorléru inicializován pomocí segistru SPCR. V tomto případě je mikrokontrolér nastaven jako Master, MSB je vysílán jako první. Hodinový signál je nastaven následovně: taktovací kmitočet je nastaven na fosc/64. Náběžná hrana je stoupající a odtoková hrana je klesající.

5.3.2 Inicializace SD karty

SD karta je napájena přímo z AVR. Proto musí mít AVR schopnost zapínat a vypínat napájení karty, jež může být použita pro šetření energií. AVR také řídí hardwarový reset. Inicializaci karty lze rozložit na 3 hlavní části (z hlediska softwaru). Nejprve je nutné označit kartu jako nevybranou (linka SS je v log. 1 - Slave Select) a poslat 80 hodinových impulsů přes linku SCK. Tato část je ukončena zasláním hodnoty 0xff přes SPI, a to desetkrát. V dalším kroku je generováno dalších 80 hodinových impulzů, ale tentokrát je na SS lince hodnota log. 0 (karta je vybrána - Slave Select). V poslední fázi je na kartu zaslán příkaz k softwarovému resetu.

5.3.3 Kontrola souborového systému

Proběhne-li inicializace karty úspěšně, je nutné nejdříve zkontrolovat první sektor na kartě (boot sektor), zda není poškozen. Tento sektor obsahuje informace o oddílech a FAT tabulkách. Byty číslo 0xef, 0x58 a 0x90 na prvních 3 pozicích sektoru označují, že se jedná o svazek informací souborového systému typu FAT. Jestli ne, partition table by měla být na pozici začínající bytem číslo 0x1be. Partition table obsahuje 4 záznamy, každý 16 bytů dlouhý (jedná se o používané oddíly na kartě). Záznamy mají adresy 0x1be, 0x1ce, 0x1de, 0x1ee.

Další byte s adresou 0x1fe (s offsetem 4) obsahuje informace o používaném souborovém systému v oddíle. Nyní je nutné zkontrolovat sektory s hodnotami 0xb nebo 0xc, zda používaný souborový systém je typu FAT32. Jestli tato kontrola selže, je nutné zaznamenat chybu v inicializačním procesu FAT. Když je první platný záznam FAT32 analyzován (0x1be), je nutné předpokládat, že ne vždy je tento záznam platný. 32bitová adresa prvního sektoru oddílu FAT32 je čtena kromě bytů 7 - 10. Protože tato adresa ukazuje na sektor s informacemi o FAT.

5.3.4 Vytvoření souboru

Před vytvořením souboru je zkontrolován adresář root, který se nachází na clusteru č. 2. První záznam v root adresáři se nazývá Volume Label Entry (záznam návěští začátku). Po tomto záznamu následují záznamy o souborech a adresářích. Pokud je root prázdný, je v něm vytvořen nový soubor. Obě FAT tabulky jsou modifikovány a je alokován 1 sektor pro tento nový soubor. Je vyplněn iniciačními daty (junk data).

5.3.5 Přidávání dat do souboru

Pokud jsou k dispozici nová data, jsou zapsána na další volný cluster na kartě. Poté jsou modifikovány obě FAT tabulky a také je upraven root podle rostoucí velikosti souboru, do kterého se zapisuje.

5.4 Vývojové nástroje

Dříve se jednočipové mikropočítače programovaly pomocí jazyka symbolických adres (assembler). Výrobci jednočipů většinou dodávali i vlastní překladače, disassemblery, debuggery, simulátory a další. Navíc řada softwarových firem poté vyráběla univerzální překladače, které se mezi sebou více či méně lišily, což mohlo způsobit problémy při kombinování několika různých zdrojových kódů.

Hlavní výhodou při programování v assembleru je absolutní kontrola nad výsledným kódem. Při dobré znalosti hardware může být výsledný kód efektivně optimalizován podle požadavků na velikost nebo rychlost. Tento postup má význam u levnějších jednočipů s malou pamětí a nízkým výkonem, kde by se výsledný kód vyšších jazyků vůbec nevešel.

Nevýhodou programování v assembleru je, že zdrojový kód je vázán na konkrétní instrukční sadu daného procesoru, či rodiny procesorů. Nelze ho tedy snadno importovat na jinou platformu, než pro kterou je určen. Assembler klade vyšší nároky na programátora, jenž musí mít perfektní znalosti konkrétní instrukční sady a musí umět daný algoritmus efektivně rozložit na elementární operace odpovídající používaným instrukcím.

5.4.1 Programovací jazyk C

Se vzrůstající rychlostí a kapacitou paměti u moderních jednočipů se stále častěji používá překladačů pro vyšší programovací jazyky. Jazyk C byl vyvinut začátkem 70. let pro účely operačního systému UNIX (Ken Thompson a Dennis Ritchie). V roce 1989 byl vyvinut standart ANSI C, zde je jeho stručná charakteristika:

- nízkoúrovňový programovací jazyk
- strukturovaný jazyk
- úsporná syntaxe
- velký soubor operátorů a moderních datových struktur
- velká efektivita kódu, rychlost téměř totožná s kódem v assembleru
- nezávislý na používané platformě

V dnešní době existuje pro každou platformu aspoň jeden překladač pro jazyk C a stejně je tomu i pro platformu AVR. Je zde možnost si vybrat z celé řady komerčních překladačů, např CodeVisionAVR C Compiler, Imagecraft ICCAVR, WinAVR (tento překladač jsem v tomto projektu použil).

5.4.2 WinAVR

Balík WinAVR obsahuje vše potřebné pro kompilaci zdrojového kódu a programování jednočipu. Zde je jeho charakteristika:

- GCC - překladač ANSI C, C++
- Binutils - nástroje pro tvorbu a práci s binárními soubory
- AVRLibC - knihovna standartních funkcí jazyka C pro AVR
- AVRdude - programátor AVR s širokou podporou hardware
- GDB - debugger
- AvarICE - podpora debuggování pro GDB
- SimulAVR - podpora simulace pro GDB
- PDF, HTLM - pro dokumentací a jiné popisy

Překlad probíhá podle schématu na obr. . Zdrojové soubory jsou nejprve zpracovány preprocesorem, který načte definované hlavičkové soubory a expanduje makra. Překladač je pak přeloží do objektového kódu (strojový kód s relativními adresami identifikátorů) a předá sestavovcímu programu (linker). Ten načte další potřebný objektový kód knihovních funkcí. Poté sestaví výsledný binární soubor, již s absolutními adresami. Program v binární nebo hexadecimální souboru je možno načíst programátorem a přes příslušný hardware (v tomto případě přes SPI) nahrát do paměti jednočipu. Linker navíc vytváří textové výpisy (.LST, .MAP). V souboru .LST je zdrojový kód prokládán řádky v assembleru tak jak jej překladač přeložil. Díky tomu je možné vyhledat konkrétní funkci (i knihovní funkci) a zjistit její zápis v assembleru. Obsahem souboru .MAP je výpis přiřazení paměťových adres identifikátorům. Linker navíc podporuje formát .ELF, který lze načít do debuggeru GDB a AVR Studia. Celý proces překladu a programování tze zautomatizovat pomocí textového skriptu makefile pro program make, jenž zajistí volání všech potřebných programů podle skriptu.

5.5 Organizace zdrojového kódu

Zdrojové kódy záznamníku jsou rozčleněny do několika knihoven - souborů podle své logické funkce. Dohromady tvoří aplikační rozhraní, které odděluje aplikaci od samotného hardware. Každá knihovna je tvořena souborem .H, kde jsou přiřazeny IO porty danému rozhraní, definice maker, datových typů, hlavičky funkcí a jejich definice. Pak je zde soubor .C, jenž obsahuje funkci main().

Pro obsluhu SD karty a SPI rozhraní, definování souborového systému FAT32 slouží knihovna SD.H. Pro obsluhu sériového rozhraní RS232 je zde knihovna USART.H.

5.6 Skript makefile

Pro snadnou kompilaci zdrojových kódů je vytvořen soubor MAKEFILE pro program MAKE. Na příkazovém řádku v adresáři se zdrojovými soubory stačí zavolat program make (bez parametrů) a ten pak provede překlad podle skriptu MAKEFILE. Výsledkem úspěšného překladu jsou souboru relativního objektového kódu .O, výpisy .LST a .MAP, binární soubor .ELF pro debugger, binární soubor .BIN a hexadecimální soubor .HEX.

Soubor MAKEFILE obsahuje několik důležitých konfiguračních řádků. Zde je jejich výpis:

- PROGNAME = BC
- OBJS = usart.o, sd.o
- MCU_TARGET = atmega32
- OPTIMIZE = 02 a další

Řádek s proměnnou PROGNAME říká, jak se bude jmenovat výsledný přeložený soubor. Proměnná OBJS obsahuje soubory relativního objektového kódu, které budou do výsledného programu zahrnuty. Typ jednočipu, pro který je výsledný program přeložen, je nutné nastavit proměnnou MCU_TARGET. Optimalizaci výsledného kódu lze ovlivnit parametry překladače gcc v proměnné OPTIMIZE.

5.7 Popis knihoven a hlaviček funkcí

V této podkapitole jsou popsány jednotlivé knihovny a hlavičky funkcí, případně proměnné.

5.7.1 Knihovna USART

V této knihovně jsou popsány funkce pro obsluhu sériového rozhraní.

- void **usartInit()**

Tato funkce inicializuje modul USART pro sériové rozhraní (nastaví jej na standartní rychlost 9,6kbps, povolí příjem).

- char **getByte()**

Čeká na příjem bytu ze sériové linky a vrací přijatý Byte.

Proměnné

- **uint8_t byte**

Příchozí Byte dat ze sériového rozhraní.

5.7.2 Knihovna SD

V této knihovně jsou popsány funkce pro obsluhu SD karty, rozhraní SPI a souborového systému FAT32.

- uint8_t **spi(uint8_t bt)**

Inicializuje SPI transakce, posílá byte bt na slave (SD karta) a vrací 8bitovou odpověď

- void **sendCommand(uint8_t cmd, uint8_t a, uint8_t b, uint8_t c, uint8_t d, uint8_t crc)**

Posílá 48bitový příkaz, 8bitový index příkazu, 32bitové parametry a b c d, 8bitová crc kontrola.

- uint8_t **cardResponse()**

Posílá hodnotu 0xff na kartu a čeká na odpověď dokud není vrácen platný byte (jiný než 0xff), pokud po 700 pokusech není přijata platná odpověď, vrací 0xff (indikuje chybu).

- uint8_t **sdInit()**

Inicializuje SD kartu (možno použít i MMC s vhodným rozhraním), při úspěchu vrací 1, jinak 0.

- uint8_t **sdReadSector(uint8_t adr)**

Čtení sektoru specifikovaného 32bitovou adresou (uložena v proměnné adr) a skladuje ji v RAM. Při úspěchu vrací 1, 0 při chybě.

- uint8_t **sdReadSectorInt(uint8_t adr)**

Vnitřní operace pro sdReadSector(uint8_t adr), zajistí aby sektor, který je v bufferu již nebyl z SD karty čten.

- uint8_t **sdFATInit()**

Inicializace souborového systému na SD kartě poté co ověří, zda se skutečně jedná o typ FAT32. Pokud selže nesmí se na kartu zapisovat (existuje zde možnost zničení dat na kartě a kartu již nebude možno používat). Při úspěchu vrací 1, jinak 0.

- uint8_t **sdWriteSector(uint8_t adr)**

Zapisuje data z bufferu v RAM do sektoru na SD kartě, specifikovaného 32bitovou adresou (v proměnné adr). Při úspěchu vrací 1, jinak 0.

- uint8_t **fileCreate()**

Přečte první sektor v root adresáři souborového systému, pokud je prázdný, vytvoří soubor AVR.TXT a přidá 512bytů iniciačních dat (junk data). Při úspěchu vrací 1, jinak 0.

- uint8_t **fileOpen()**

Sdružuje inicializační procedury. Při úspěchu (úspěšná inicializace SD, úspěšná kontrola FAT, root adresář je prázdný a úspěšné vytvoření souboru) vrací 1, jinak 0.

- uint8_t **fileAppend()**

Přidává 512bytů dat z bufferu v RAM do souboru vytvořeného funkcí fileCreate(). Nejprve zapíše data, poté aktualizuje obě FAT tabulky a nakonec upraví velikost souboru.

Proměnné

- **uint32_t buff_sector = 0xffffffff**

Globální proměnná obsahující adresu sektoru zkopírovaného do bufferu, při každé změně obsahu bufferu se musí hodnota této proměnné na neplatnou hodnotu (např. 0x00), před čtením sektoru pomocí sdReadSector se tato proměnná ověří pro zamezení nadbytečných čtení z karty.

- **uint32_t LBA_offset**

Offset datové oblasti karty (soubory), nesmí se používat když je 0.

- **uint32_t LBA_offset_a**

Offset primární FAT, nesmí se používat když je 0.

- **uint32_t LBA_offset_b**

Offset sekundární FAT, nesmí se používat když je 0.

- **uint32_t max_sector**

Poslední platný sektor datové oblasti, před přidáváním dat do souboru se musí zkontrolovat zda není překřížený.

- **uint32_t last_cluster_written**

Číslo clusteru, jenž byl zapsán do souboru, je zkontrolován a aktualizován při přidávání dat do souboru.

- **uint32_t *buffer**

Ukazatel na 512 bytů v RAM, dočasně skladovaných.

- **#define CMD0 0x40**

Hexadecimální hodnoty příkazů používaných v SPI módu.

6 Závěr

Na závěr bych zhodnotil a popsal výsledky práce na tomto projektu. V první etapě jsem prostudoval specifikace k použitému mikrokontroléru AVR ATmega32, jeho strukturu, porty, registry, atd. Dále pak specifikace k sériovému rozhraní RS232 a také k převodníku úrovní MAX232, potřebného k připojení RS232 k mikrokontroléru. Po seznámení s procesorem jsem sestavil jednoduchý prototyp s připojenou LED na DPS. Do tohoto prototypu byl zaveden program, jenž měl střídavě rozsvěcovat a zhasínat LED. Sloužil pro osvojení postupů při programování a ověřování funkčnosti AVR.

Poté jsem k procesoru připojil sériové rozhraní přes převodník úrovní MAX232. Osazení a oživení DPS s rozhraním RS232 proběhlo naprosto bez problémů. Části programu pro obsluhu periferií jsem naprogramoval flefibilně (hlavně z hlediska přenosové rychlosti), takže stačí stačí změnit hodnoty v hlavičkových souborech. Hlavní výhodou je, že pro tento projekt lze použít i jiný procesor typu ATmega s dostatečnou kapacitou paměti. Postačí změnit definice v hlavičkových souborech a typ cílového procesoru v makefile.

Po zhotovení modulu s rozhraním RS232 na DPS jsem začal studovat specifikace SD karty. Specifikace ohledně napájení, fyzického a elektrického rozhraní, způsoby čtení a zápisu dat. Dále potom specifikace systémových požadavků (kapacita paměti, přenosová rychlost, frekvence hodin, copyright, atd.). Značnou pozornost jsem věnoval rozhraní pro přenos dat z a na SD kartu (SPI) a také způsobům čtení a zápisu dat. Po prostudování těchto specifikací jsem k již zhotovenému modulu připojil SD kartu. Poté jsem se věnoval programovému vybavení pro komunikaci MCU s SD kartou přes rozhraní SPI. Konkrétně inicializace rozhraní SPI, posílání příkazů a další potřebné procedury. Tyto procedury jsem zavedl do programu.

Dále jsem se věnoval studiu souborového systému typu FAT, jeho struktuře, indikace a ověřování typu FAT, metodám zápisu a čtení dat v rámci souborového systému.

V poslední etapě jsem se věnoval programovým metodám pro zápis a čtení dat na konkrétní SD kartě. Velké komplikace nastaly při realizace zápisu dat přijatých ze sériového rozhraní. V tomto projektu se data zapisují po sektorech v rámci jednoho souboru, proto je nutné zjistit adresu posledního sektoru kde byly zapsány data. Kdyby došlo k chybnému určení sektoru mohlo by dojít k přepsání dat nebo dokonce k zápisu dat mimo stanovený soubor.

7 Literatura

[1] Katalogové listy AVR ATmega32,
http://www.atmel.com/dyn/resources/prod_documents/doc2503.pdf

[2] Katalogové listy MAX232,
http://www.datasheetcatalog.org/data/texasinstruments/max232.pdf

[3] Specifikace SD karty,
http://www.sdcard.org/developers/tech/sdcard/pls/Simplified_Physical_Layer_Spec.pdf

[4] Specifikace a používání SD karty,
http://elm-chan.org/docs/mmc/mmc_e.html

[5] Specifikace souborového systému FAT (specifikace od Microsoftu),
http://www.microsoft.com/whde/system/platform/firmware/fatgendown.mspx
soubor fatgen103.doc

[6] Popis struktury souborového systému FAT,
http://www.win.tue.nl/ãeb/linux/fs/fat/fat-1.html

[7] Popis USART,
http://noel.feld.cvut.cz/vyu/scs/prezentace2008/USART/#mozTocId206734

[8] Specifikace AVRlib (Procyon AVRlib),
http://www.mil.üfl.edu/čhrisarnold/components/microcontrollerBoard/AVR/avrlib

[9] Specifikace RS232 (EIA232),
http://www.camiresearch.com/Data_Com_Basic/RS232_standart.html

Seznam tabulek

Seznam obrázků

8 Obsah přiloženého CD

textová část práce: $\backslash text\backslash$

zdrojové kódy: $\backslash src\backslash$

používané datasheety: $\backslash datasheet\backslash$

schémata zařízení: $\backslash schema\backslash$